## Two-way Binding with NgModel

When developing data entry forms, we often want to both display a data property and update that property when the user makes changes.

The [(ngModel)] two-way data binding syntax makes that easy. Here's an example:

<input [(ngModel)]="currentHero.firstName">

[()] = BANANA IN A BOX

To remember that the parentheses go inside the brackets, visualize a banana in a box.

Alternatively, we can use the canonical prefix form:

<input bindon-ngModel="currentHero.firstName">

We could have achieved the same result with separate bindings to the <input> element's value property and input event. That’s however cumbersome.

<input [value]="currentHero.firstName" (input)="currentHero.firstName=$event.target.value" >

That ngModel directive hides these onerous details behind its own ngModel input and ngModelChange output properties.

<input [ngModel]="currentHero.firstName" (ngModelChange)="currentHero.firstName=$event">

*The ngModel input property sets the element's value property and the ngModelChange output property listens for changes to the element's value. The details are specific to each kind of element and therefore the NgModel directive only works for elements, such as the input text box, that are supported by a ControlValueAccessor. We can't apply [(ngModel)] to our custom components until we write a suitable value accessor.*

We shouldn't have to mention the data property twice. Angular should be able to capture the component’s data property and set it with a single declaration — which it can with the [( )] syntax:

<input [(ngModel)]="currentHero.firstName">

### Syntactic suger[(x)]

[(x)] is just syntactic sugar for a property binding and an event binding:

[x]="someParentProperty" (xChange)="someParentProperty=$event"

To achive the following code:

<my-comp [(myText)]="testString"></my-comp>

In the component for my-comp must have an myText property and an myTextChange property(which is a EventEmitter).

**export class** MyComp {  
 @Input() **myText**: **string**;  
 @Output() **myTextChange**: EventEmitter<**string**> = **new** EventEmitter();

*//Notify parent of changes, whenever the value of myText changes, emit an event.* onChange(newMyText:**string**) {  
 **this**.**myTextChange**.emit(newMyText);  
 }  
}

### Aliasing input/output properties

# Words

**cumbersome** 英 **['kʌmbəsəm]**  美 **['kʌmbərsəm]**  **adj.笨重的；不方便的**

**cumber** 英 ['kʌmbə]   美 ['kʌmbə]  v.**拖累；妨碍** n.**妨碍；累赘**

**onerous** 英 ['əʊnərəs]  美 ['ɑːnərəs] adj.**繁重的**